Government Smart Card Interoperability Specification v2.1 (NISTIR 6887 – 2003 Edition) Basic Services Interface Java Binding

Conformance Test Instantiation, Verification, and Reporting Scenarios

FINAL DRAFT

Alan Goldfine

October 8, 2004

This document contains the conformance test instantiation, verification, and reporting scenarios for the methods comprising the Java language binding of the Basic Services Interface of version 2.1 of the Government Smart Card Interoperability Specification (GSC-IS), as contained in NIST Interagency Report 6887 – 2003 Edition.

The 23 sections of this document correspond to the 23 sections in the Java Binding Conformance Test Assertions document.

Appendix A contains the list of constant variables (symbolic constants) used in this document. A constant variable is indicated by a leading underscore (e.g., _PIN).

Appendix B contains a description of a set of three cards that would be sufficient for the testing of candidate BSI implementations, when using test suites built according to these scenarios.

1. gscBsiUtilAcquireContext()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0100.

2. There is no authenticated session established with any container on the connected card

3. There exists a container on the connected card for which
   - readTagListACR is PIN Protected
   - the value of the PIN is _PIN
   - the container is represented by AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. There exists a Vector strctAuthenticator0100 with one element, the BSIAuthenticator object BSIAuthenticator0100. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

6. There exists a Vector tagArrayA0100.

Test for Assertion 1.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.1".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0100.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or BSI_TERMINAL_AUTH (a BSIException is thrown, with BSIException.getErrorCode returning BSI_TERMINAL_AUTH).

2. An authenticated session is established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_OK or BSI_TERMINAL_AUTH, then verify that an authenticated session has indeed been established with the target container.

   Make a tagArrayA0100 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that an authenticated session had been established.
   Status: Test 1.1 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that an authenticated session had not been established.
   Status: Test 1.1 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters returned an incorrect code.
   Status: Test 1.1 Failed."

Test for Assertion 1.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.2".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard /= hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0100.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario*:*

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.1.2.1.

   Verify that an authenticated session has not been established with the target container:

   Make a tagArrayA0100 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad handle has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.2 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.2 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilAcquireContext() called with a bad handle returned an incorrect code.
   Status: Test 1.2 Failed."

Test for Assertion 1.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.3".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100

- AID == _badGSCAID (GSC) or _badCACAID (CAC)
- strctAuthenticator == strctAuthenticator0100.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.1.3.1 using
   - hCard == hCard0100.

   Print
   "gscBsiUtilAcquireContext() called with a bad AID value has been verified.
   Status: Test 1.3 Passed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_AID, then print
   "gscBsiUtilAcquireContext() called with a bad AID value returned an incorrect code.
   Status: Test 1.3 Failed."

Test for Assertion 1.4

The method is tested using an authentication method that is not available on the card.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Declare an array of bytes challenge0104.

3. (Pre) Make a challenge0104 == gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Construct a Vector strctAuthenticator0104 with the element BSIAuthenticator0104. The BSIAuthenticator object BSIAuthenticator0104 has fields
   - accessMethodType == BSI_AM_XAUTH
   - keyIDOrReference == _keyIDOrReference1
   - authValue == challenge0104.

Note: The gscBsiGetChallenge() call was issued because the call itself may be required by an implementation prior to an gscBsiUtilAcquireContext() call; the actual content of challenge0104 in BSIAuthenticator0104 is unimportant.

5. (Pre) Print "Testing of Assertion 1.4".

6. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0104.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACR_NOT_AVAILABLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACR_NOT_AVAILABLE).

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_ACR_NOT_AVAILABLE, then:

   Perform Test for Assertion 9.1.4.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a tagArrayA0100 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with an authentication method that is not available on the card has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.4 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with an authentication method that is not available on the card has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.4 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_ACR_NOT_AVAILABLE, then print

"gscBsiUtilAcquireContext() called with an authentication method that is not available on the card returned an incorrect code.
Status: Test 1.4 Failed."

Test for Assertion 1.5

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 1.6

The method is tested using a bad authenticator.

Verification Goal:
To verify Expected Result 2: "An authenticated session is not established with the target container."

Perform this verification by issuing a call to gscBsiGcReadTagList().

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Construct a Vector strctAuthenticator0106 with the element BSIAuthenticator0106. The BSIAuthenticator object BSIAuthenticator0106 has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _badAuthValue.

3. (Pre) Print "Testing of Assertion 1.6".
   -
4. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0106.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AUTH (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AUTH).

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_AUTH, then:

   Perform Test for Assertion 9.1.6.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a tagArrayA0100 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad authenticator has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had been not established.
   Status: Test 1.6 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad authenticator has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.6 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_AUTH, then print
   "gscBsiUtilAcquireContext() called with a bad authenticator returned an incorrect code.
   Status: Test 1.6 Failed."

Test for Assertion 1.7

The method is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Remove the connected card from the reader.

3. (Pre) Print "Testing of Assertion 1.7".

4. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0100.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilAcquireContext() called with the connected card removed has been verified.
   Status: Test 1.7 Passed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilAcquireContext() called with the connected card removed returned an incorrect code.
   Status: Test 1.7 Failed."

Test for Assertion 1.8

The method is tested with a blocked PIN.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Construct a Vector strctAuthenticator0108 with one element, the BSIAuthenticator object BSIAuthenticator0108. BSIAuthenticator0108 has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference2
   - authValue0108 == _goodAuthValue2.

3. (Pre) Make N gscBsiUtilAcquireContext() calls to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0108

   where N is the maximum number of incorrect PIN tries allowed by the implementation.

4. (Pre) Print "Testing of Assertion 1.8".

5. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0100.

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_PIN_BLOCKED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_PIN_BLOCKED).

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_PIN_BLOCKED, then:

   Perform Test for Assertion 9.1.8.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a tagArrayA0100 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.8 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.8 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_PIN_BLOCKED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN returned an incorrect code.
   Status: Test 1.8 Failed."

2. gscBsiUtilConnect()

Starting State for Each Test:

1. There exists a card reader, whose name is represented by the String readerName0200, available to the candidate implementation.

2. There exists an int hCard0200.

Test for Assertion 2.1

The method is tested using valid parameters, with a good card inserted into a specified reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader readerName0200.

3. (Pre) Print "Testing of Assertion 2.1".

4. Make an hCard0200 = gscBsiUtilConnect() call to the SPS, with
   - readerName == readerName0200.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown)
   - hCard0200 == a valid handle.

2. The card is connected, with handle hCard0200, to the reader readerName0200.

Perform this verification by issuing a call to gscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_OK, then verify that the card is indeed connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0200.

   Case 1.1: If gscBsiUtilGetCardStatus() returns the code BSI_OK, then print
   "gscBsiUtilConnect() called with valid parameters has been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card had been connected.
   Status: Test 2.1 Passed."

Case 1.2: If gscBsiUtilGetCardStatus() does not return the code BSI_OK, then print
"gscBsiUtilConnect() called with valid parameters has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card had not been connected.
Status: Test 2.1 Failed."

Case 2: If the gscBsiUtilConnect() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called with valid parameters with a good card inserted into a specified reader returned an incorrect code.
Status: Test 2.1 Failed."

Test for Assertion 2.2

The method is tested using valid parameters, with a good card inserted into a non-specified reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader readerName0200.

3. (Pre) Print "Testing of Assertion 2.2".

4. Make an hCard0200 = gscBsiUtilConnect() call to the SPS, with
   - readerName == "".

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown)
   - hCard0200 == a valid handle.

2. The card is connected, with handle hCard0200, to the first available reader.

Perform this verification by issuing a call to gscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_OK, then verify that the card is indeed connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0200.

   Case 1.1: If gscBsiUtilGetCardStatus() returns the code BSI_OK, then print
   "gscBsiUtilConnect() called with valid parameters has been verified because a subsequent call to

gscBsiUtilGetCardStatus() was successful, indicating that the card had been connected.
Status: Test 2.2 Passed."

Case 1.2: If gscBsiUtilGetCardStatus() does not return the code BSI_OK, then print
"gscBsiUtilConnect() called with valid parameters has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card had not been connected.
Status: Test 2.2 Failed."

Case 2: If the gscBsiUtilConnect() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called with valid parameters with a good card inserted into a non-specified reader returned an incorrect code.
Status: Test 2.2 Failed."

Test for Assertion 2.3

The method is tested using a bad reader name.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().
2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader readerName0200.
3. (Pre) Print "Testing of Assertion 2.3".
4. Make an hCard0200 = gscBsiUtilConnect() call to the SPS, with
   - readerName == _badReaderName.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_UNKNOWN_READER (a BSIException is thrown, with BSIException.getErrorCode returning BSI_UNKNOWN_READER).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_UNKNOWN_READER, then print
   "gscBsiUtilConnect() called with a bad reader name has been verified.
   Status: Test 2.3 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_UNKNOWN_READER, then print
   "gscBsiUtilConnect() called with a bad reader name returned an incorrect code.
   Status: Test 2.3 Failed."

Test for Assertion 2.4

The method is tested with no card in the reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Manually ensure that there is no valid card in the particular reader represented by the String readerName0200.

3. (Pre) Print "Testing of Assertion 2.4".

4. Make an hCard0200 = gscBsiUtilConnect() call to the SPS, with
   - readerName == the String readerName0200.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_ABSENT (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_ABSENT).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_CARD_ABSENT, then print
   "gscBsiUtilConnect() called with no card in the reader has been verified.
   Status: Test 2.4 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_CARD_ABSENT, then print
   "gscBsiUtilConnect() called with no card in the reader returned an incorrect code.
   Status: Test 2.4 Failed."

Test for Assertion 2.5

The method is tested using a bad inserted card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that does not claim conformance to the GSC-IS into the reader readerName0200.

3. Make an hCard0200 = gscBsiUtilConnect() call to the SPS, with
   - readerName == the String readerName0200.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_ABSENT or BSI_UNKNOWN_ERROR (a BSIException is thrown, with BSIException.getErrorCode returning either BSI_CARD_ABSENT or BSI_UNKNOWN_ERROR).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_CARD_ABSENT or the code BSI_UNKNOWN_ERROR, then print
   "gscBsiUtilConnect() called with a bad inserted card has been verified.
   Status: Test 2.5 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_CARD_ABSENT or the code BSI_UNKNOWN_ERROR, then print
   "gscBsiUtilConnect() called with a bad inserted card returned an incorrect code.
   Status: Test 2.5 Failed."

3. gscBsiUtilDisconnect()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a particular reader available to the candidate implementation.

2. The card is not connected.

3. The name of the reader is represented by the String readerName0300.

4. There exists an int hCard0300.

5. Make an hCard0300 = gscBsiUtilConnect() call to the SPS, with
   - readerName == readerName0300.

Test for Assertion 3.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. (Pre) Print "Testing of Assertion 3.1".

3. Make a gscBsiUtilDisconnect() call to the SPS, with
   - hCard == hCard0300.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown).

2. The card is disconnected.

Perform this verification by issuing a call to GscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_OK, then verify that the card is indeed disconnected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0300.

   Case 1.1: If gscBsiUtilGetCardStatus() does not return the code BSI_OK, then print
   "gscBsiUtilDisconnect() called with valid parameters has been verified because a subsequent call to gscBsiUtilGetCardStatus() was not successful, indicating that the card was no longer connected.
   Status: Test 3.1 Passed."

Case 1.2: If gscBsiUtilGetCardStatus() returns the code BSI_OK, then print
"gscBsiUtilDisconnect() called with valid parameters has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card was still connected.
Status: Test 3.1 Failed."

Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_OK, then print
"gscBsiUtilDisconnect() called with valid parameters returned an incorrect code.
Status: Test 3.1 Failed."

Test for Assertion 3.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. Make a gscBsiUtilDisconnect() call to the SPS, using
   - hCard /= hCard0300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

2. (Pre) Print "Testing of Assertion 3.2".

3. The card is still connected.

Perform this verification by issuing a call to GscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.3.2.1 using
   - hCard == hCard0300.

   Verify that the card is still connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0300.

   Case 1.1: If gscBsiUtilGetCardStatus() returns the code BSI_OK, then print
   "gscBsiUtilDisconnect() called with a bad handle has been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card was still connected.

Status: Test 3.2 Passed."

Case 1.2: If gscBsiUtilGetCardStatus() does not return the code BSI_OK, then print
"gscBsiUtilDisconnect() called with a bad handle has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card was no longer connected.
Status: Test 3.2 Failed."

Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_BAD_HANDLE, then print
"gscBsiUtilDisconnect() called with a bad handle returned an incorrect code.
Status: Test 3.2 Failed."

Test for Assertion 3.3

The method is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. (Pre) Remove the connected card from the reader.

3. (Pre) Print "Testing of Assertion 3.3".

4. Make a gscBsiUtilDisconnect() call to the SPS, with
   - hCard == hCard0300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilDisconnect() called with the connected card removed has been verified.
   Status: Test 3.3 Passed."

   Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilDisconnect() called with the connected card removed returned an incorrect code.
   Status: Test 3.3 Failed."

4. gscBsiUtilBeginTransaction()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0400.

Test for Assertion 4.1

The method is tested as a blocking transaction call, with no existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.1".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is no existing transaction lock. Assertion 4.1 of gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.1.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == true.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

2. If the return code is BSI_OK, then a transaction is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_OK, then verify that a transaction has indeed been started:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1.1: If gscBsiUtilEndTransaction() returns the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, has been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
   Status: Test 4.1 Passed."

   Case 1.2: If gscBsiUtilEndTransaction() does not return the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.
   Status: Test 4.1 Failed."

   Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction() is not supported.
   Status: Test 4.1 Not Supported."

   Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, returned an incorrect code.
   Status: Test 4.1 Failed."

Test for Assertion 4.2

The method is tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.2".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
"It cannot be assured that there is no existing transaction lock. Assertion 4.2 of gscBsiUtilBeginTransaction() cannot be tested."
End Test for Assertion 4.2.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == false.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

2. If the return code is BSI_OK, then a transaction is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_OK, then verify that a transaction has indeed been started:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1.1: If gscBsiUtilEndTransaction() returns the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, has been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
   Status: Test 4.2 Passed."

   Case 1.2: If gscBsiUtilEndTransaction() does not return the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.
   Status: Test 4.2 Failed."

Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction() is not supported.
Status: Test 4.2 Not Supported."

Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, returned an incorrect code.
Status: Test 4.2 Failed."

Test for Assertion 4.3

The method is tested as a non-blocking transaction call, with another application having established a transaction lock, using valid parameters.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 4.4

The method is tested as a blocking transaction call, with another application having established a transaction lock, using valid parameters.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 4.5

The method is tested as a non-blocking transaction call using valid parameters, after the current application establishes a transaction lock.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.5".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400.
   - blType == false.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

Case 2: If the gscBsiUtilBeginTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
"It cannot be assured that there is an existing transaction lock. Assertion 4.5 of gscBsiUtilBeginTransaction() cannot be tested."
End Test for Assertion 4.5.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == false.

Verification Goal:
To verify the Expected Results:
1. The call returns the return code BSI_NOT_TRANSACTED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NOT_TRANSACTED).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.4.5.1 using
   - hCard == hCard0400.

   Print
   "gscBsiUtilBeginTransaction(), tested as a non-blocking transaction call, using valid parameters, after the current application has established a transaction lock, has been verified.
   Status: Test 4.5 Passed."

   Case 2: If the gscBsiUtilBeginTransaction() call does not return the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a non-blocking transaction call, using valid parameters, after the current application has established a transaction lock, returned an incorrect code.
   Status: Test 4.5 Failed."

Test for Assertion 4.6

The method is tested as a blocking transaction call using valid parameters, after the current application establishes a transaction lock.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.6".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with

- hCard == hCard0400.
- blType == false.

Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

Case 2: If the gscBsiUtilBeginTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
"It cannot be assured that there is an existing transaction lock. Assertion 4.6 of gscBsiUtilBeginTransaction() cannot be tested."
End Test for Assertion 4.6.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == true.

Verification Goal:
To verify the Expected Results:
1. The call returns the return code BSI_NOT_TRANSACTED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NOT_TRANSACTED).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.4.6.1 using
   - hCard == hCard0400.

   Print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, using valid parameters, after the current application has established a transaction lock, has been verified.
   Status: Test 4.6 Passed."

   Case 2: If the gscBsiUtilBeginTransaction() call does not return the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, using valid parameters, after the current application has established a transaction lock, returned an incorrect code.
   Status: Test 4.6 Failed."

Test for Assertion 4.7

The method is tested as a blocking transaction call, with no existing transaction lock, with a bad handle.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.7".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is no existing transaction lock. Assertion 4.7 of gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.7.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard /= hCard0400
   - blType == true.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

2. No transaction lock is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.4.7.1 using
   - hCard == hCard0400.

   Verify that no transaction lock has been established:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1.1: If gscBsiUtilEndTransaction() returns the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, has been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.

Status: Test 4.7 Passed."

Case 1.2: If gscBsiUtilEndTransaction() returns the code BSI_OK, then print
"gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
Status: Test 4.7 Failed."

Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction() is not supported.
Status: Test 4.7 Not Supported."

Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, returned an incorrect code.
Status: Test 4.7 Failed."

5. gscBsiUtilEndTransaction()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0500.

Test for Assertion 5.1

The method is tested with an existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.1".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500.
   - blType == true.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is an existing transaction lock. Assertion 5.1 of gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 5.1.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0500.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

2. If the return code is BSI_OK, then the previously existing transaction lock is ended.

Perform this verification by issuing a call to gscBsiUtilBeginTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then verify that the existing transaction has indeed ended:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500
   - blType == true.

   Case 1.1: If gscBsiUtilBeginTransaction() returns the code BSI_OK, then print
   "gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters has been verified because a subsequent call to gscBsiUtilBeginTransaction() was successful, indicating that the previous transaction had been ended.
   Status: Test 5.1 Passed."

   Case 1.2: If gscBsiUtilEndTransaction() does not return the code BSI_OK, then print
   "gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters has not been verified because a subsequent call to gscBsiUtilBeginTransaction() was unsuccessful, indicating that the previous transaction had not been ended.
   Status: Test 5.1 Failed."

   Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() is not supported.
   Status: Test 5.1 Not Supported."

   Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters returned an incorrect code.
   Status: Test 5.1 Failed."

Test for Assertion 5.2

The method is tested with no existing transaction lock.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.2".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0500.

Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
"It cannot be assured that there is no existing transaction lock. Assertion 5.2 of gscBsiUtilBeginTransaction() cannot be tested."
End Test for Assertion 5.2.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0500.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_NOT_TRANSACTED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NOT_TRANSACTED) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.5.2.1 using
   - hCard == hCard0500.

   Print
   "gscBsiUtilEndTransaction() called with no existing transaction lock and valid parameters has been verified.
   Status: Test 5.2 Passed."

   Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() is not supported.
   Status: Test 5.2 Not Supported."

   Case 3: If the gscBsiUtilEndTransaction() call does not return the code BSI_NOT_TRANSACTED or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() called with no existing transaction lock and valid parameters returned an incorrect code.
   Status: Test 5.2 Failed."

Test for Assertion 5.3

The method is tested with an existing transaction lock, using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.3".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500.
   - blType == true.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is an existing transaction lock. Assertion 5.3 of gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 5.3.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard /= hCard0500.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_SPSSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_SPSSERVICE).

2. If the return code is BSI_BAD_HANDLE, then the previously existing transaction lock remains in effect.

Perform this verification by issuing a call to gscBsiUtilBeginTransaction().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.5.3.1 using
   - hCard == hCard0500.

   Verify that the existing transaction is still in effect:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500
   - blType == true.

   Case 1.1: If gscBsiUtilBeginTransaction() returns the code BSI_NOT_TRANSACTED, then print

"gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle has been verified because a subsequent call to gscBsiUtilBeginTransaction() was unsuccessful, indicating that the previous transaction had not been ended.
Status: Test 5.3 Passed."

Case 1.2: If gscBsiUtilEndTransaction() returns the code BSI_OK, then print
"gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle has not been verified because a subsequent call to gscBsiUtilBeginTransaction() was successful, indicating that that the previous transaction had been ended.
Status: Test 5.3 Failed."

Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilEndTransaction() is not supported.
Status: Test 5.3 Not Supported."

Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle returned an incorrect code.
Status: Test 5.3 Failed."

6. gscBsiUtilGetVersion()

Test for Assertion 6.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 6.1".

2. Make a String version0600 = gscBsiUtilGetVersion() call to the SPS.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown)
   - version0600 == the BSI implementation version of the SPS.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetVersion() call returns the code BSI_OK, then manually inspect the String version0600.
   Case 1.1: If version0600 represents the BSI implementation version of the SPS, then print
   "gscBsiUtilGetVersion() called with valid parameters has been verified by inspection.
   Status: Test 6.1 Passed."

   Case 1.2: If version0600 does not represent the BSI implementation version of the SPS, then print
   "gscBsiUtilGetVersion()called with valid parameters has not been verified by inspection.
   Status: Test 6.1 Failed."

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_OK, then print
   "gscBsiUtilGetVersion() called with valid parameters returned an incorrect code.
   Status: Test 6.1 Failed."

7. gscBsiUtilGetCardProperties()

Starting State for each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0700.

2. There exists a CardProperties object cardProps0700, with fields
   - protected int cardCapability0700
   - protected byte[] CCCUniqueID0700.

Test for Assertion 7.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Print "Testing of Assertion 7.1".

3. Make a cardProps0700 = gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then
   - cardCapability0700 == one of the recognized bitwise masks identifying the provider of the connected card
   - CCCUniqueID0700 == the Card Capability Container ID.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_OK, then manually inspect cardCapability0700 and CCCUniqueID0700.

   Case 1.1: If cardCapability0700 is one of
   - 00000001
   - 00000002
   - 00000004

   and CCCUniqueID0700 is the Card Capability Container ID, then print
   "gscBsiUtilGetCardProperties() called with valid parameters has been verified by inspection.
   Status: Test 7.1 Passed."

Case 1.2: If either cardCapability0700 is not one of the above masks or CCCUniqueID0700 is not the Card Capability Container ID, then print
"gscBsiUtilGetCardProperties() called with valid parameters has not been verified by inspection.
Status: Test 7.1 Failed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.1 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() called with valid parameters returned an incorrect code.
Status: Test 7.1 Failed."

Test for Assertion 7.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Print "Testing of Assertion 7.2".

3. Make a cardProps0700 = gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard /= hCard0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.7.2.1 using
   - hCard == hCard0700.

   Print
   "gscBsiUtilGetCardProperties () called with a bad handle has been verified.
   Status: Test 7.2 Passed."

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.2 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties () call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties () called with a bad handle returned an incorrect code.
Status: Test 7.2 Failed."

Test for Assertion 7.3

The method is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Remove the connected card from the reader.

3. (Pre) Print "Testing of Assertion 7.3".

4. Make a cardProps0700 = gscBsiUtilGetCardProperties() call to the SPS, with
   - hCard == hCard0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardProperties ()called with the connected card removed has been verified.
   Status: Test 7.3 Passed."

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() is not supported.
   Status: Test 7.3 Not Supported."

   Case 3: If the gscBsiUtilGetCardProperties () call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties () called with the connected card removed returned an incorrect code.
   Status: Test 7.3 Failed."

Test for Assertion 7.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

8. gscBsiUtilGetCardStatus()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0800.

Test for Assertion 8.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Print "Testing of Assertion 8.1".

3. Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0800.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
   "gscBsiUtilGetCardStatus() called with valid parameters has been verified.
   Status: Test 8.1 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
   "gscBsiUtilGetCardStatus() called with valid parameters returned an incorrect code.
   Status: Test 8.1 Failed."

Test for Assertion 8.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Print "Testing of Assertion 8.2".

3. Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard /= hCard0800.

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.8.2.1 using
   - hCard == hCard0800.

   Print
   "gscBsiUtilGetCardStatus() called with a bad handle has been verified.
   Status: Test 8.2 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilGetCardStatus() called with a bad handle returned an incorrect code.
   Status: Test 8.2 Failed."

Test for Assertion 8.3

The method is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Remove the connected card from the reader.

3. (Pre) Print "Testing of Assertion 8.3".

4. Make a gscBsiGetCardStatus() call to the SPS, with
   - hCard == hCard0800.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardStatus() called with the connected card removed has been verified.
   Status: Test 8.3 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardStatus() called with the connected card removed returned an incorrect code.
   Status: Test 8.3 Failed.

9. gscBsiUtilGetExtendedErrorText()

Test for Assertion 9.X.Y.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 9.X.Y.1".

2. Make a String errorText0900 = gscBsiUtilGetExtendedErrorText call to the SPS, using
   - hCard == hCard0X00 (X<9) or hCardX00 (X>9).

Verification Goal:

To verify the Expected Results:

1. The call returns
   - either the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_TEXT_AVAILABLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_TEXT_AVAILABLE)
   - if BSI_OK is the code returned, then ErrorText0900 == an extended error message
   - if BSI_NO_TEXT_AVAILABLE is the code returned, then ErrorText0900== "".

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetExtendedErrorText() call returns the code BSI_OK, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters has been verified (text returned).
   Status: Test 9.X.Y.1 Passed."

   Case 2: If the gscBsiGetExtendedErrorText() call returns the code BSI_NO_TEXT_AVAILABLE, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters has been verified (no text available).
   Status: Test 9.X.Y.1 Passed."

   Case 3: If the gscBsiGetExtendedErrorText() call returns a code other than BSI_OK or BSI_NO_TEXT_AVAILABLE, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters returned an incorrect code.
   Status: Test 9.X.Y.1 Failed."

10. gscBsiUtilGetReaderList()

Test for Assertion 10.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 10.1".

2. Make a Vector vReaderList1000 = gscBsiUtilGetReaderList() call to the SPS.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown)
   - vReaderList1000 == a Vector of Strings representing the available readers.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetReaderList() call returns the code BSI_OK, then manually inspect the Strings constituting the returned Vector vReaderList1000.

   Case 1.1: If the collection of Strings does represent the collection of available readers, then print
   "gscBsiUtilGetReaderList() called with valid parameters has been verified by inspection.
   Status: Test 10.1 Passed."

   Case 1.2: If the collection of Strings does not represent the available readers, then print
   "gscBsiUtilGetReaderList() called with valid parameters has not been verified by inspection.
   Status: Test 10.1 Failed."

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_OK, then print
   "gscBsiUtilGetReaderList() called with valid parameters returned an incorrect code.
   Status: Test 10.1 Failed."

11. gscBsiUtilPassthru()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1100.

2. There is declared an array of bytes cardResponse1100.

Test for Assertion 11.1

The method is tested with valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Print "Testing of Assertion 11.1".

3. Make a cardResponse1100 = gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - cardCommand == _goodCardCommand.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown)
   - cardResponse1100 == an array of bytes containing the APDU response from the connected card.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_OK, then:

   Case 1.1: If cardResponse1100 == one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() called with valid parameters has been verified.
   Status: Test 11.1 Passed."

   Case 1.2: If cardResponse1100 /= one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() called with valid parameters has not been verified.
   Status: Test 11.1 Failed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_OK, then print
   "gscBsiUtilPassthru() called with valid parameters returned an incorrect code.
   Status: Test 11.1 Failed."

Test for Assertion 11.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Print "Testing of Assertion 11.2".

3. Make a cardResponse1100 = gscBsiUtilPassthru() call to the SPS, using
   - hCard /= hCard1100
   - cardCommand == _goodCardCommand.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.11.2.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() called with a bad handle has been verified.
   Status: Test 11.2 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilPassthru() called with a bad handle returned an incorrect code.
   Status: Test 11.2 Failed."

Test for Assertion 11.3

The method is tested using a bad cardCommand.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) There exists an int cardCommand1103 == _badCardCommand.

3. (Pre) Print "Testing of Assertion 11.3".

4. Make a cardResponse1100 = gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - cardCommand == _badCardCommand.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_PARAM).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.11.3.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() called with a bad card command has been verified.
   Status: Test 11.3 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilPassthru () called with a bad card command returned an incorrect code.
   Status: Test 11.3 Failed."

Test for Assertion 11.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 11.5

The method is tested with a card that has been removed.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().
2. (Pre) Remove the connected card from the reader.
3. (Pre) Print "Testing of Assertion 11.5".
4. Make a cardResponse1100 = gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - cardCommand == goodCardCommand.

Verification Goal:
To verify the Expected Results:
1. The call returns

- the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilPassthru() called with the connected card removed has been verified.
   Status: Test 11.5 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilPassthru () called with the connected card removed returned an incorrect code.
   Status: Test 11.5 Failed."

12. gscBsiUtilReleaseContext()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1200.

2. There exists a target container on the connected card for which
   - readTagListACR is PIN Protected
   - the value of the PIN is _PIN
   - the container is represented by AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

3. There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

4. There exists a Vector strctAuthenticator1200 with one element, the BSIAuthenticator object BSIAuthenticator1200. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1

5. There exists a Vector tagArrayA1200.

6. There is an authenticated session established with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1200
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1200.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with step 3 of the Instantiation Scenario.

   Case 2: If gscBsiUtilGetCardStatus() does not return the code BSI_OK, then print
   "A session cannot be established.
   gscBsiUtilReleaseContext() cannot be tested".
   End current test.

Test for Assertion 12.1

The method is tested with valid parameters.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.1".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200

- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown).

2. There is no longer an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_OK, then verify that there is no longer an authenticated session established with the target container:

   Make a tagArrayA1200 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters has been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
   Status: Test 12.1 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.1 Failed."

   Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters returned an incorrect code.
   Status: Test 12.1 Failed."

Test for Assertion 12.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.2".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using

- hCard /= hCard1200
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

2. There continues to be an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.12.2.1 using
   - hCard == hCard1200.

   Verify that there continues to be an authenticated session established with the target container:

   Make a tagArrayA1200 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad handle has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.2 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
   Status: Test 12.2 Failed."

Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_BAD_HANDLE, then print
"gscBsiUtilReleaseContext() called with a bad handle returned an incorrect code.
Status: Test 12.2 Failed."

Test for Assertion 12.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.3".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - AID == _badGSCAID (GSC) or _badCACAID (CAC).

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID).

2. There continues to be an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.12.3.1 using
   - hCard == hCard1200.

   Verify that there continues to be an authenticated session established with the target container:

   Make a tagArrayA1200 = gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.3 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad AID value D has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
   Status: Test 12.3 Failed."

   Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_BAD_AID, then print
   "gscBsiUtilReleaseContext() called with a bad AID value returned an incorrect code.

Status: Test 12.3 Failed."

Test for Assertion 12.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 12.5

The method is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.5".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. (Pre) Remove the connected card from the reader.

4. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilReleaseContext() called with the connected card removed has been verified.
   Status: Test 12.5 Passed."

   Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilReleaseContext() called with the connected card removed returned an incorrect code.
   Status: Test 12.5 Failed."

13. gscBsiGcDataCreate()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator1300 with one element, the BSIAuthenticator object BSIAuthenticator1300. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. There is declared an array of bytes dValue1300.

Test for Assertion 13.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1301.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - there does not exist a data item in the container with tag == _newTag1301
   - the container can accommodate the data item _newDvalue1301.

4. (Pre) Print "Testing of Assertion 13.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1301
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.1 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.1.

6. Make a gscBsiGcDataCreate() call to the SPS, using

- hCard == hCard1301
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- tag == _newTag1301
- value == _newDvalue1301.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then the data value _newDvalue1301 is stored, with tag == _newTag1301, in the target container.

3. No other changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_OK, then verify that that the specified data value has indeed been stored, with the specified tag, in the target container.

   Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1301
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1301.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1300 == _newDvalue1301

   then print
   "gscBsiGcDataCreate() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() was successful, indicating that the specified data value was correctly created.
   Status: Test 13.1 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1300 /= _newDvalue1301

   then print
   " gscBsiGcDataCreate() called with valid parameters has been not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value was not correctly created.
   Status: Test 13.1 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return the code BSI_OK, then print

"gscBsiGcDataCreate() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.1 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.1 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_OK or the return code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with valid parameters returned an incorrect code.
Status: Test 13.1 Failed."

Test for Assertion 13.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1302.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - there does not exist a data item in the container with tag == _newTag1302
   - the container can accommodate the data item _newDvalue1302.

4. (Pre) Print "Testing of Assertion 13.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1302
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.2 of gscBsiGcDataCreate() cannot be tested".

End Test for Assertion 13.2.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard /= hCard1302
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1302
   - value == _newDvalue1302.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.13.2.1 using
   - hCard == hCard1302.

   Verify that that the specified data value was not stored, with the specified tag, in the target container:

   Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1302
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1302.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes
   - BSI_BAD_TAG
   - BSI_IO_ERROR

   then print
   "gscBsiGcDataCreate() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
   Status: Test 13.2 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK

   then print
   "gscBsiGcDataCreate() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
   Status: Test 13.2 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_OK
- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called with a bad handle _HANDLE has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.2 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.2 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad handle returned an incorrect code.
Status: Test 13.2 Failed."

Test for Assertion 13.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1303.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - there does not exist a data item in the container with tag == _newTag1303
   - the container can accommodate the data item _newDvalue1303.

4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 13.3".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1303

- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- strctAuthenticator == strctAuthenticator1300.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.3 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 13.3.

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1303
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - tag == _newTag1303
   - value == _newDvalue1303.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.13.3.1 using
   - hCard == hCard1303.

   Verify that that the specified data value was not stored, with the specified tag, in the target container:

   Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1303
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1303.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes
   - BSI_BAD_TAG
   - BSI_IO_ERROR

   then print
   "gscBsiGcDataCreate() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
   Status: Test 13.3 Passed."

Case 1.2: If the gscBsiGcReadValue() call returns
- the code BSI_OK

then print
"gscBsiGcDataCreate() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.3 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_OK
- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.3 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.3 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad AID value returned an incorrect code.
Status: Test 13.3 Failed."

Test for Assertion 13.4

The method is tested using a bad parameter.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1304.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 13.4".
   -

5. (Pre) Establish an authenticated session with the target container on the card:

Make a gscBsiUtilAcquireContext() call to the SPS, using
- hCard == hCard1304
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- strctAuthenticator == strctAuthenticator1300.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.4 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 13.4.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1304
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _invalidTag
   - value == _newDvalue1304.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_PARAM) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.13.4.1 using
   - hCard == hCard1304.

   Print
   "gscBsiGcDataCreate() called with a bad parameter has been verified.

   Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() is not supported.
   Status: Test 13.4 Not Supported."

   Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() called with a bad parameter returned an incorrect code.
   Status: Test 13.4 Failed."

Test for Assertion 13.5

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 13.6

The method is tested using a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1306.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - there does not exist a data item in the container with tag == _newTag1306
   - the container can accommodate the data item _newDvalue1306.

4. (Pre) Print "Testing of Assertion 13.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1306
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.6 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.6.

6. (Pre) Remove the connected card from the reader.

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1306
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1306
   - value == _newDvalue1306.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcDataCreate() called with the connected card removed has been verified.
   Status: Test 13.6 Passed."

   Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() is not supported.
   Status: Test 13.6 Not Supported."

   Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() called with the connected card removed returned an incorrect code.
   Status: Test 13.6 Failed."

Test for Assertion 13.7

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1307.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container can accommodate the data item _newDvalue1307.

4. (Pre) There does not exist a data item in the target container with tag == _newTag1307.

5. (Pre) Ensure that there is no authenticated session with the target container:

Make a gscBsiUtilReleaseContext() call to the SPS, using
- hCard == hCard1307
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

6. (Pre) Print "Testing of Assertion 13.7".

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1307
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1307
   - value == _newDvalue1307.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.13.7.1 using
   - hCard == hCard1307.

   Verify that that the specified data value was not stored, with the specified tag, in the target container:

   Establish an authenticated session with the target container:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1307
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1.1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then:

   Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1307
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1307.

   Case 1.1.1: If the gscBsiGcReadValue() call returns either of the codes
   - BSI_BAD_TAG

• BSI_IO_ERROR
then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.7 Passed."

Case 1.1.2: If the gscBsiGcReadValue() call returns
• the code BSI_OK
then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.7 Failed."

Case 1.1.3: If the gscBsiGcReadValue() call does not return any of the codes
• BSI_OK
• BSI_BAD_TAG
• BSI_IO_ERROR
then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.7 Undetermined."

Case 1.2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session could not be established. Assertion 13.7 of gscBsiGcDataCreate() cannot be verified.
Status: Test 13.7 Undetermined."
End Test for Assertion 13.7.

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.7 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 13.7 Failed."

Test for Assertion 13.8

The method is tested using a too-large data value.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1308.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - the target container contains one data item, which comprises the entire available space of the container. The tag for this data item is _existingTagFull.

4. (Pre) Print "Testing of Assertion 13.8".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1308
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.8 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.8.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1308
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - tag == _newTag1308
   - value == _newDvalue1308.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_NO_MORE_SPACE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_MORE_SPACE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_NO_MORE_SPACE, then:

Perform Test for Assertion 9.13.8.1 using
- hCard == hCard1308.

Verify that that the specified data value was not stored, with the specified tag, in the target container:

Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
- hCard == hCard1308
- AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
- tag == _newTag1308.

Case 1.1: If the gscBsiGcReadValue() call returns either of the codes
- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called using a too-large data value has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.8 Passed."

Case 1.2: If the gscBsiGcReadValue() call returns
- the code BSI_OK

then print
" gscBsiGcDataCreate() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.8 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_OK
- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.8 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.8 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_NO_MORE_SPACE or the return code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called using a too-large data value returned an incorrect code.
Status: Test 13.8 Failed."

Test for Assertion 13.9

The method is tested using the tag of a data item that already exists.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1309.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - there exists a data item in the container with tag == _existingTag1309 and value _existingDvalue1309
   - the container can accommodate the data item _newDvalue1309.

4. (Pre) Print "Testing of Assertion 13.9".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1309
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1300.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.9 gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.9.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1309
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1309
   - value == _newDvalue1309.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_TAG_EXISTS (a BSIException is thrown, with BSIException.getErrorCode returning BSI_TAG_EXISTS) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_TAG_EXISTS, then:

   Perform Test for Assertion 9.13.9.1 using
   - hCard == hCard1309.

   Verify that that the specified data value was not stored, with the specified tag, in the target container:

   Make a dValue1300 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1309
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1309.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1300 == _existingDvalue1309

   then print
   "gscBsiGcDataCreate() called using the tag of a data item that already exists has been verified because a subsequent call to gscBsiGcReadValue() indicated that no writing to the target container had occurred.
   Status: Test 13.9 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1300 /= _existingDvalue1309

   then print
   "gscBsiGcDataCreate() called using the tag of a data item that already exists has not been verified because a subsequent call to gscBsiGcReadValue() indicated that writing to the target container had occurred.
   Status: Test 13.9 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return the code
   - BSI_OK

   then print
   "gscBsiGcDataCreate() called using the tag of a data item that already exists has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
   Status: Test 13.9 Undetermined."

   Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() is not supported.
   Status: Test 13.9 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_TAG_EXISTS or the return code BSI_NO_CARDSERVICE, then print

"gscBsiGcDataCreate() called using the tag of a data item that already exists returned an incorrect code.
Status: Test 13.9 Failed."

14. gscBsiGcDataDelete()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator1400 with one element, the BSIAuthenticator object BSIAuthenticator1400. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. There is declared an array of bytes dValue1400.

Test for Assertion 14.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1401.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains at least one data item, for which
     - the tag is == _existingTag1401
     - the value is _existingDvalue1401.

4. (Pre) Print "Testing of Assertion 14.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1401
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1400.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.1 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.1.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1401

- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- tag == _existingTag1401.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then there is no longer a data item with the tag == _existingTag1401 stored in the target container.

3. No other changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_OK, then verify that that the specified data item has indeed been deleted from the target container.

   Make a dValue1400 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1401
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1401.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_BAD_TAG

   then print
   "gscBsiGcDataDelete() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() did not find the deleted data item.
   Status: Test 14.1 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK

   then print
   "gscBsiGcDataDelete() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item was not deleted.
   Status: Test 14.1 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
   - BSI_OK
   - BSI_BAD_TAG

   then print
   "gscBsiGcDataDelete() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
   Status: Test 14.1 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.1 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_OK or the code BSI_NO_CARSERVICE, then print
"gscBsiGcDataDelete() called with valid parameters returned an incorrect code.
Status: Test 14.1 Failed."

Test for Assertion 14.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1402.

3. (Pre) There exists a target container on the connected card with the following properties
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains at least one data item, for which
     - the tag is == _existingTag1402
     - the value is _existingDvalue1402.

4. (Pre) Print "Testing of Assertion 14.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1402
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1400.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.2 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.2.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard /= hCard1402
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- tag == _existingTag1402.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.14.2.1 using
   - hCard == hCard1402.

   Verify that that the specified data value was not deleted from the target container:

   Make a dValue1400 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1402
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1402.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1400 == _existingDvalue1402

   then print
   "gscBsiGcDataDelete() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
   Status: Test 14.2 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_BAD_TAG

   or
   - the code BSI_OK
   - dValue1400 /= _existingDvalue1402

   then print
   "gscBsiGcDataDelete() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had been deleted or otherwise changed.
   Status: Test 14.2 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
   - BSI_OK

- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.2 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.2 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called with a bad handle returned an incorrect code.
Status: Test 14.2 Failed."

Test for Assertion 14.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1403.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains at least one data item, for which
     - the tag is == _existingTag1403
     - the value is _existingDvalue1403.

4. There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 14.3".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1403
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1400.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 14.3 of gscBsiGcDataDelete() cannot be tested".
End Test for Assertion 14.3.

7. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1403
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - tag == _existingTag1403.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.14.3.1 using
   - hCard == hCard1403.

   Verify that that the specified data value was not deleted from the target container:

   Make a dValue1400 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1403
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1403.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1400 == _existingDvalue1403

   then print
   "gscBsiGcDataDelete() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
   Status: Test 14.3 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_BAD_TAG

   or
   - the code BSI_OK
   - dValue1400 /= _existingDvalue1403

   then print

"gscBsiGcDataDelete() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had been deleted or otherwise changed.
Status: Test 14.3 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.3 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.3 Not Supported."

Case 2: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSWRVICE, then print
"gscBsiGcDataDelete() called with a bad AID value returned an incorrect code.
Status: Test 14.3 Failed."

Test for Assertion 14.4

The method is tested using a bad tag.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1404.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container does not contain a data item for which the tag is == _newTag1404.

4. (Pre) Print "Testing of Assertion 14.4".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1404
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- strctAuthenticator == strctAuthenticator1400.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 14.4 of gscBsiGcDataDelete() cannot be tested".
End Test for Assertion 14.4.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1404
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1404.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_TAG (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_TAG) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.14.4.1 using
   - hCard == hCard1404.

   Print
   "gscBsiGcDataDelete() called with a bad tag has been verified.
   Status: Test 14.4 Passed."

   Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() is not supported.
   Status: Test 14.4 Not Supported."

   Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() called with a bad tag returned an incorrect code.
   Status: Test 14.4 Failed."

Test for Assertion 14.5

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 14.6

The method is tested using a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1406.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains at least one data item, for which
     - the tag is == _existingTag1406
     - the value is _existingDvalue1406.

4. (Pre) Print "Testing of Assertion 14.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1406
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1400.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.6 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.6.

6. (Pre) Remove the connected card from the reader.

7. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1406
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1406.

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcDataDelete() called with the connected card removed has been verified.
   Status: Test 14.6 Passed."

   Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() is not supported.
   Status: Test 14.6 Not Supported."

   Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiGcDataDelete() called with the connected card removed returned an incorrect code.
   Status: Test 14.6 Failed."

Test for Assertion 14.7

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1407.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains at least one data item, for which
     - the tag is == _existingTag1407
     - the value is _existingDvalue1407.

4. (Pre) Ensure that there is no authenticated session with the target container.

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1407
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

5. (Pre) Print "Testing of Assertion 14.7".
   -
6. Make a gscBsiGcDataDelete() call to the SPS, using

- hCard == hCard1407
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- tag == _existingTag1407.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.14.7.1 using
   - hCard == hCard1407.

   Verify that that the specified data value was not deleted from the target container:

   Establish an authenticated session with the target container

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1407
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1400.

   Case 1.1: If the gscBsiUtilAcquireContext call returns the code BSI_OK, then

   Make a dValue1400 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1407
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1407.

   Case 1.1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1400 == _existingDvalue1407

   then print
   "gscBsiGcDataDelete() called without fulfilling the applicable ACR has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted. **Status**: Test 14.7 Passed."

   Case 1.1.2: If the gscBsiGcReadValue() call returns

- the code BSI_BAD_TAG

or

- the code BSI_OK
- dValue1400 /= _existingDvalue1407

then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data item had been deleted or otherwise changed.
Status: Test 14.7 Failed."

Case 1.1.3: If the gscBsiGcReadValue() call does not return either of the codes

- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.7 Undetermined."

Case 1.2: If the gscBsiUtilAcquireContext call does not return the code BSI_OK, then print
"A session cannot be established.
gscBsiUtilDataDelete cannot be verified.
Status: Test 14.7 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.7 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 14.7 Failed."

15. gscBsiGcGetContainerProperties()

Starting state for each Test:

1. There exists a ContainerProperties object containerProps1500 with fields
   - GCacr strctGCacr1500, with fields
     - BSIAcr createACR1500, with fields
       - int createACRType1500
       - int[] createKeyIDOrReference1500
       - int createAuthNb1500
       - int createACRID1500
     - BSIAcr deleteACR1500, with fields
       - int deleteACRType1500
       - int[] deleteKeyIDOrReference1500
       - int deleteAuthNb1500
       - int deleteACRID1500
     - BSIAcr readTagListACR1500, with fields
       - int readTagListACRType1500
       - int[] readTagListKeyIDOrReference1500
       - int readTagListAuthNb1500
       - int readTagListACRID1500
     - BSIAcr readValueACR1500, with fields
       - int readValueACRType1500
       - int[] readValueKeyIDOrReference1500
       - int readValueAuthNb1500
       - int readValueACRID1500
     - BSIAcr updateValueACR1500, with fields
       - int updateValueACRType1500
       - int[] updateValueKeyIDOrReference1500
       - int updateValueAuthNb1500
       - int updateValueACRID1500
   - GCContainerSize strctContainerSizes1500, with fields
     - int maxNbDataItems1500
     - int maxValueStorageSize1500
   - String containerVersion1500.

Test for Assertion 15.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1501.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN

- the content of the keyID or reference array == _keyIDOrReference1
- number of access methods logically combined in the ACR == 1
- ACRID == 0

• the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 15.1".

5. Make a containerProps1500 == gscBsiGcGetContainerProperties() call to the SPS, using
   • hCard == hCard1501
   • AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   • the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then the variables of strctGCacr1500 are correctly set to indicate access control conditions for all operations.

Note: We are not currently verifying the returned values of strctContainerSizes1500 and containerVersion1500.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_OK, then:
   Case 1.1: If
   - each of createACRType1500, deleteACRType1500, readTagListACRType1500, readValueACRType1500, and updateValueACRType1500 == BSI_ACR_PIN

   and
   - the content of each of createKeyIDOrReference1500, deleteKeyIDOrReference1500, readTagListKeyIDOrReference1500, readValueKeyIDOrReference1500, and updateKeyIDOrReference1500 == _keyIDOrReference1

   and
   - each of createAuthNB1500, deleteAuthNB1500, readTagListAuthNB1500, readValueAuthNB1500, and updateValueAuthNB1500 == 1

   and
   - each of createACRID1500, deleteACRID1500, readTagListACRID1500, readValueACRID1500, and updateValueACRID1500 == 0

   then print
   "gscBsiGcGetContainerProperties() called with valid parameters has been verified.
   Status: Test 15.1 Passed."

Case 1.2: If
- any of createACRType1500, deleteACRType1500, readTagListACRType1500, readValueACRType1500, and updateValueACRType1500 /= BSI_ACR_PIN

or
- the content of any of createKeyIDOrReference1500, deleteKeyIDOrReference1500, readTagListKeyIDOrReference1500, readValueKeyIDOrReference1500, and updateKeyIDOrReference1500 /= _keyIDOrReference1

or
- any of createAuthNB1500, deleteAuthNB1500, readTagListAuthNB1500, readValueAuthNB1500, and updateValueAuthNB1500 /= 1

or
- any of createACRID1500, deleteACRID1500, readTagListACRID1500, readValueACRID1500, and updateValueACRID1500 /= 0

then print
"gscBsiGcGetContainerProperties() called with valid parameters has not been verified.
Status: Test 15.1 Failed."

Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() is not supported.
Status: Test 15.1 Not Supported."

Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() called with valid parameters returned an incorrect code.
Status: Test 15.1 Failed."

Test for Assertion 15.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1502.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1

- number of access methods logically combined in the ACR == 1
- ACRID == 0

- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 15.2".

5. Make a containerProps1500 == gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard /= hCard1502
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.15.2.1 using
   - hCard == hCard1502.

   Print
   "gscBsiGcGetContainerProperties() called with a bad handle has been verified.
   Status: Test 15.2 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.2 Not Supported."

   Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() called with a bad handle returned an incorrect code.
   Status: Test 15.2 Failed."

Test for Assertion 15.3

The method is tested using a bad AID value.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1503.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 15.3".

6. Make a containerProps1500 == gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard == hCard1503
   - AID == _badGSCAID (GSC) or _badCACAID (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.15.3.1 using
   - hCard == hCard1503.

   Print
   "gscBsiGcGetContainerProperties() called with a bad AID value has been verified.
   Status: Test 15.3 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.3 Not Supported."

Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() called with a bad AID value returned an incorrect code.
Status: Test 15.3 Failed."

Test for Assertion 15.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 15.5

The method is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1505.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Remove the connected card from the reader.

6. (Pre) Print "Testing of Assertion 15.5".

5. Make a containerProps1500 == gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard == hCard1505
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcGetContainerProperties() called with the connected card removed has been verified.
   Status: Test 15.5 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.5 Not Supported."

   Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() called with the connected card removed returned an incorrect code.
   Status: Test 15.5 Failed."

16. gscBsiGcReadTagList()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator1600 with one element, the BSIAuthenticator object BSIAuthenticator1600. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. There is declared an array of shorts tagListArray1600.

Test for Assertion 16.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1601.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a collection of 27 data items whose tags are:
     - _existingTag1309
     - _existingTag1310
     - _existingTag1401
     - _existingTag1402
     - _existingTag1403
     - _existingTag1404
     - _existingTag1405
     - _existingTag1406
     - _existingTag1407
     - _existingTag1408
     - _existingTag1701
     - _existingTag1702
     - _existingTag1703
     - _existingTag1704
     - _existingTag1706
     - _existingTag1707
     - _existingTag1709
     - _existingTag1710
     - _existingTag1801
     - _existingTag1802
     - _existingTag1803
     - _existingTag1804
     - _existingTag1805

- o _existingTag1806
- o _existingTag1807
- o _existingTag1808
- o _existingTag1809.

4. (Pre) Print "Testing of Assertion 16.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1601
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1600.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.1 of gscBsiGcReadTagList() cannot be tested".
   End Test for Assertion 16.1.

6. Make a tagListArray1600 == gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1601
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then tagListArray1600 == an array containing the list of tags for the target container.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_OK, then:

   Case 1.1: If tagListArray1600 contains precisely the 27 elements:
   - o _existingTag1309
   - o _existingTag1310
   - o _existingTag1401
   - o _existingTag1402
   - o _existingTag1403
   - o _existingTag1404
   - o _existingTag1405
   - o _existingTag1406
   - o _existingTag1407
   - o _existingTag1408
   - o _existingTag1701

- _existingTag1702
- _existingTag1703
- _existingTag1704
- _existingTag1706
- _existingTag1707
- _existingTag1709
- _existingTag1710
- _existingTag1801
- _existingTag1802
- _existingTag1803
- _existingTag1804
- _existingTag1805
- _existingTag1806
- _existingTag1807
- _existingTag1808
- _existingTag1809

then print
"gscBsiGcReadTagList() with a called with valid parameters has been verified.
Status: Test 16.1 Passed."

Case 1.2: If tagListArray1600 does not contain precisely the 27 elements:

- _existingTag1309
- _existingTag1310
- _existingTag1401
- _existingTag1402
- _existingTag1403
- _existingTag1404
- _existingTag1405
- _existingTag1406
- _existingTag1407
- _existingTag1408
- _existingTag1701
- _existingTag1702
- _existingTag1703
- _existingTag1704
- _existingTag1706
- _existingTag1707
- _existingTag1709
- _existingTag1710
- _existingTag1801
- _existingTag1802
- _existingTag1803
- _existingTag1804
- _existingTag1805
- _existingTag1806
- _existingTag1807
- _existingTag1808
- _existingTag1809

then print
" gscBsiGcReadTagList () called with valid parameters has not been verified.
Status: Test 16.1 Failed."

Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiGcReadTagList() is not supported.
Status: Test 16.1 Not Supported."

Case 3: If the gscBsiGcReadTagList () call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() called with valid parameters returned an incorrect code.
Status: Test 16.1 Failed."

Test for Assertion 16.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1602.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 16.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1602
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1600.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.2 of gscBsiGcReadTagList() cannot be tested".
   End Test for Assertion 16.2.

6. Make a tagListArray1600 == gscBsiGcReadTagList() call to the SPS, using
   - hCard /= hCard1602
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.16.2.1 using
   - hCard == hCard1602.

   Print
   "gscBsiGcReadTagList() called with a bad handle has been verified.
   Status: Test 16.2 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.2 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARSERVICE, then print
   "gscBsiGcReadTagList() called with a bad handle returned an incorrect code.
   Status: Test 16.2 Failed."

Test for Assertion 16.3

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 16.4

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1604.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN

- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 16.4".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1604
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1600.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.4 of gscBsiGcReadTagList() cannot be tested".
   End Test for Assertion 16.4.

7. Make a tagListArray1600 == gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1604
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID), or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.16.4.1 using
   - hCard == hCard1604.

   Print
   "gscBsiGcReadTagList() called with a bad AID value has been verified.
   Status: Test 16.4 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.4 Not Supported."

Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() called with a bad AID value returned an incorrect code.
Status: Test 16.4 Failed."

Test for Assertion 16.5

The method is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1605.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 16.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1605
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1600.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 5.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.5 of gscBsiGcReadTagList() cannot be tested".
   End Test for Assertion 16.5.

6. (Pre) Remove the connected card from the reader.

7. Make a tagListArray1600 == gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1605
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning

BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_AID, then print
   "gscBsiGcReadTagList() called with the connected card removed has been verified.
   Status: Test 16.5 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.5 Not Supported."

   Case 2: If the gscBsiGcReadTagList() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() called with the connected card removed returned an incorrect code.
   Status: Test 16.5 Failed."

Test for Assertion 16.6

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1606.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1606
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

5. (Pre) Print "Testing of Assertion 16.6".

6. Make a tagListArray1600 == gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1606
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.16.6.1 using
   - hCard == hCard1606.

   Print
   "gscBsiGcReadTagList() called without fulfilling the applicable ACR has been verified.
   Status: Test 16.6 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.6 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() called without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 16.6 Failed."

17. gscBsiGcReadValue()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator1700 with one element, the BSIAuthenticator object BSIAuthenticator1700. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. There is declared an array of bytes dValue1700.

Test for Assertion 17.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1701.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item such that
     - the tag of the data item is == _existingTag1701
     - the value of the data item is == _existingDvalue1701.

4. (Pre) Print "Testing of Assertion 17.1".
   -

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1701
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1700.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.1 of gscBsiGcReadValue() cannot be tested".
   End Test for Assertion 17.1.

6. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using

- hCard == hCard1701
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- tag == _existingTag1701.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then dValue1700 == _existingDvalue1701.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_OK, then:
   Case 1.1: If dValue1700 == _existingDvalue1701, then print "gscBsiGcReadValue() called with valid parameters has been verified.
   Status: Test 17.1 Passed."

   Case 1.2: If dValue1700 /= _existingDvalue1701, then print "gscBsiGcReadValue() called with valid parameters has not been verified.
   Status: Test 17.1 Failed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.1 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() called with valid parameters returned an incorrect code.
   Status: Test 17.1 Failed."

Test for Assertion 17.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1702.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN

- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- the container contains a data item whose tag is == _existingTag1702.

4. (Pre) Print "Testing of Assertion 17.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1702
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1700.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.2 of gscBsiGcReadValue() cannot be tested".
   End Test for Assertion 17.2.

6. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using
   - hCard /= hCard1702
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1702.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.17.2.1 using
   - hCard == hCard1702.

   Print
   "gscBsiGcReadValue() called with a bad handle has been verified.
   Status: Test 17.2 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.2 Not Supported."

Case 2: If the gscBsiGcReadValue() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() called with a bad handle returned an incorrect code.
Status: Test 17.2 Failed."

Test for Assertion 17.3

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 17.4

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1704.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item whose tag is == _existingTag1704
   - the value of the data item is == _existingDvalue1704.

4. (Pre) There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 17.4".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1704
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1700.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print

"A session cannot be established. Assertion 17.4 cannot be tested".
End Test for Assertion 17.4.

7. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1704
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - tag == _existingTag1704.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.17.4.1 using
   - hCard == hCard1704.

   Print
   "gscBsiGcReadValue() called with a bad AID value has been verified."
   Status: Test 17.4 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.4 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() called with a bad AID value returned an incorrect code.
   Status: Test 17.4 Failed."

Test for Assertion 17.5

The method is tested using a bad tag.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1705.

3. (Pre) There exists a target container on the connected card with the following properties:

- the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- the container does not contain a data item for which the tag == _newTag1705.

4. (Pre) Print "Testing of Assertion 17.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1705
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1700.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.5 cannot be tested".
   End Test for Assertion 17.5.

6. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1705
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1705.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_TAG (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_TAG) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.17.5.1 using
   - hCard == hCard1705.

   Print
   "gscBsiGcReadValue() called with a bad tag has been verified.
   Status: Test 17.5 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.

Status: Test 17.5 Not Supported."

Case 2: If the gscBsiGcReadValue() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() called with a bad tag returned an incorrect code.
Status: Test 17.5 Failed."

Test for Assertion 17.6

The method is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1706.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item whose tag is == _existingTag1706
   - the value of the data item is == _existingDvalue1706.

4. (Pre) Print "Testing of Assertion 17.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1706
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1700.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.6 of gscBsiGcReadValue() cannot be tested".
   End Test for Assertion 17.6.

6. (Pre) Remove the connected card from the reader.

7. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1706
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1706.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcReadValue() called with the connected card removed has been verified.
   Status: Test 17.6 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.6 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() called with the connected card removed returned an incorrect code.
   Status: Test 17.6 Failed."

Test for Assertion 17.7

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1707.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC))
   - the container contains a data item whose tag is == _existingTag1707
   - the value of the data item is == _existingDvalue1701.

4. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1707

- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

5. (Pre) Print "Testing of Assertion 17.7".

6. Make a dValue1700 == gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1707
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1707.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.17.7.1 using
   - hCard == hCard1707.

   Print
   "gscBsiGcReadValue() called without fulfilling the applicable ACR has been verified.
   Status: Test 17.7 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.7 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() called without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 17.7 Failed."

18. gscBsiGcUpdateValue()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator1800 with one element, the BSIAuthenticator object BSIAuthenticator1800. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. There is declared an array of bytes dValue1800.

Test for Assertion 18.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1801.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item for which
     - the tag == _existingTag1801
     - the value == existingDvalue1801.

4. (Pre) Print "Testing of Assertion 18.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1801
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.1 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.1.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1801
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- tag == _existingTag1801
- dValue == _newDvalue1801.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then the data item in the target container for which the tag == _existingTag1801 now has the value _newDvalue1801.

3. No other changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_OK, then verify that that the specified data item now has the value _newDvalue1801.

   Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1801
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1801.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 == _newDvalue1801

   then:

   Print
   "gscBsiGcUpdateValue() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been correctly updated.
   Status: Test 18.1 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 /= _newDvalue1801

   then:

   Print
   "gscBsiGcUpdateValue() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item was not updated correctly.
   Status: Test 18.1 Failed."

   Case 1.3: If the gscBsiGcReadValue() call returns any code other than

- BSI_OK

then:

Print
"gscBsiGcUpdateValue() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.1 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.1 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() with valid parameters returned an incorrect code.
Status: Test 18.1 Failed."

Test for Assertion 18.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1802.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item for which
     - the tag == _existingTag1802
     - the value == existingDvalue1802.

4. (Pre) Print "Testing of Assertion 18.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1802
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 18.2 of gscBsiGcUpdateValue() cannot be tested".
End Test for Assertion 18.2.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard /= hCard1802
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1802
   - dValue == _newDvalue1802.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.18.2.1 using
   - hCard == hCard1802.

   Verify that that the specified data value in the target container was not changed:

   Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1802
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1802.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 == _existingDvalue1802

   then print
   "gscBsiGcUpdateValue() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
   Status: Test 18.2 Passed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 /= _existingDvalue1802

   then print

"gscBsiGcUpdateValue() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() found that the data value had been updated or otherwise changed.
Status: Test 18.2 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return the code
- BSI_OK

then print
"gscBsiGcUpdateValue() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.2 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.2 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called with a bad handle returned an incorrect code.
Status: Test 18.2 Failed."

Test for Assertion 18.3

The method is tested using a bad AID value.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1803.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item for which
     - the tag == _existingTag1803
     - the value == existingDvalue1803.

4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 18.3".

6. (Pre) Establish an authenticated session with the target container on the card:

Make a gscBsiUtilAcquireContext() call to the SPS, using
- hCard == hCard1803
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- strctAuthenticator == strctAuthenticator1800.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 18.3 of gscBsiGcUpdateValue() cannot be tested".
End Test for Assertion 18.3.

7. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1803
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - tag == _existingTag1803
   - dValue == _newDvalue1803.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.18.3.1 using
   - hCard == hCard1803.

   Verify that that the specified data value in the target container was not changed:

   Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1803
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1803.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 == existingDvalue1803

   then print
   "gscBsiGcUpdateValue() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.

Status: Test 18.3 Passed."

Case 1.2: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- dValue1800 /= existingDvalue1803

then print
"gscBsiGcUpdateValue() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() found that the data value had been updated or otherwise changed.
Status: Test 18.3 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return the code
- BSI_OK

then print
"gscBsiGcUpdateValue() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.3 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.3 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called with a bad AID value returned an incorrect code.
Status: Test 18.3 Failed."

Test for Assertion 18.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 18.5

The method is tested using a bad parameter.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1805.

3. (Pre) There exists a target container on the connected card with the following properties:

- the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 18.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1805
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.5 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.5.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1805
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _invalidTag
   - dValue == _newDvalue1805.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_PARAM) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.18.5.1 using
   - hCard == hCard1805.

   Print
   "gscBsiGcUpdateValue() called using a bad parameter has been verified.
   Status: Test 18.5 Passed."

   Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcUpdateValue() is not supported.
   Status: Test 18.5 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called with a bad parameter returned an incorrect code.
Status: Test 18.5 Failed."

Test for Assertion 18.6

The method is tested using a bad tag.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1806.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container does not contain a data item for which the tag == _newTag1806.

4. (Pre) Print "Testing of Assertion 18.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1806
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.6 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.6.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1806
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1806
   - dValue == _newDvalue1806.

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_TAG (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_TAG) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.18.6.1 using
   - hCard == hCard1806.

   Verify that that a data item with the specified tag was not created:

   Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1806
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _newTag1806.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK

   then print
   "gscBsiGcUpdateValue() called with a bad tag has not been verified because a subsequent call to gscBsiGcReadValue() found a data item with the specified tag.
   Status: Test 18.6 Failed."

   Case 1.2: If the gscBsiGcReadValue() call returns
   - the code BSI_BAD_TAG

   then print
   "gscBsiGcUpdateValue() called with a bad tag has been verified because a subsequent call to gscBsiGcReadValue() did not find a data item with the specified tag.
   Status: Test 18.6 Passed."

   Case 1.3: If the gscBsiGcReadValue() call does not return the code
   - BSI_OK or BSI_BAD_TAG

   then print
   "gscBsiGcUpdateValue() called with a bad tag has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
   Status: Test 18.6 Undetermined."

   Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcUpdateValue() is not supported.

Status: Test 18.6 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called with a bad tag returned an incorrect code.
Status: Test 18.6 Failed."

Test for Assertion 18.7

The method is tested using a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1807.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item for which the tag == _existingTag1807.

4. (Pre) Print "Testing of Assertion 18.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1807
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.7 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.7.

6. (Pre) Remove the connected card from the receiver.

7. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1807
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1807
   - dValue == _newDvalue1807.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcReadValue() called with the connected card removed has been verified.
   Status: Test 18.7 Passed."

   Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcUpdateValue() is not supported.
   Status: Test 18.7 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcUpdateValue() called with the connected card removed returned an incorrect code.
   Status: Test 18.7 Failed."

Test for Assertion 18.8

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1808.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - the container contains a data item for which
     - the tag == _existingTag1808
     - the value == existingDvalue1808.

4. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1808
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

5. (Pre) Print "Testing of Assertion 18.8".

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1808
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1808
   - dValue == _newDvalue1808.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.18.8.1 using
   - hCard == hCard1808.

   Verify that that the specified data value in the target container was not changed:

   Establish an authenticated session with the target container:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1808
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1.1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then

   Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1808
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - tag == _existingTag1808.

   Case 1.1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - dValue1800 == _existingDvalue1808

   then print
   "gscBsiGcDataUpdate() called without fulfilling the applicable ACR has been verified because a

subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been updated.
Status: Test 18.8 Passed."

Case 1.1.2: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- dValue1800 /= _existingDvalue1808

then print
"gscBsiGcDataUpdate() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the data item had been changed.
Status: Test 18.8 Failed."

Case 1.1.3: If the gscBsiGcReadValue() call does not return the code
- BSI_OK

then print
"gscBsiGcDataUpdate() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.8 Undetermined."

Case 1.2: If the gscBsiUtilAcquireContext call does not return the code BSI_OK, then print
"A session with the target container could not be established. gscBsiUtilDataUpdate cannot be verified.
Status: Test 18.8 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.8 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 18.8 Failed."

Test for Assertion 18.9

The method is tested using a too-large data value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard1809.

3. (Pre) There exists a target container on the connected card with the following properties:

- the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
- the container contains one data item, which comprises the entire available space of the container
  - the tag for this data item is == existingTagFull
  - the value of this data item is == _existingDvalueFull.

4. (Pre) Print "Testing of Assertion 18.9".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1809
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator1800.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.9 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.9.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1809
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - tag == _existingTagFull
   - dValue == _tooBigDvalue.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_NO_MORE_SPACE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_MORE_SPACE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_NO_MORE_SPACE, then:

   Perform Test for Assertion 9.18.9.1 using
   - hCard == hCard1809.

Verify that that the specified data value in the target container was not changed:

Make a dValue1800 = gscBsiGcReadValue() call to the SPS, using
- hCard == hCard1809
- AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
- tag == _existingTagFull.

Case 1.1: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- dValue1800 == _existingDvalueFull

then print
"gscBsiGcUpdateValue() called using a too-large data value has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
Status: Test 18.9 Passed."

Case 1.2: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- dValue1800 /= _existingDvalueFull

then print
"gscBsiGcUpdateValue() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() found that the data value had been updated or otherwise changed.
Status: Test 18.9 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return the code
- BSI_OK

then print
"gscBsiGcUpdateValue() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.9 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.9 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_NO_MORE_SPACE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called using a too-large data value returned an incorrect code.
Status: Test 18.9 Failed."

19. gscBsiGetChallenge()

Starting State for Each Test:

1. challenge1900 is an array of bytes.

Test for Assertion 19.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetChallenge().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1901.

3. (Pre) There exists a target container on the connected card with AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 19.1".

5. Make a challenge1900 == gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1901
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then challenge1900 == an array of bytes containing the random challenge returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_OK, then manually inspect the array challenge1900.
   Case 1.1: If challenge1900 contains a valid random challenge, then print
   "gscBsiGetChallenge() called with valid parameters has been verified by inspection.
   Status: Test 19.1 Passed."

   Case 1.2: If challenge1900 does not contain a valid random challenge, then print
   "gscBsiGetChallenge() called with valid parameters has not been verified by inspection.
   Status: Test 19.1 Failed."

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.
Status: Test 19.1 Not Supported."

Case 3: If the gscBsiGetChallenge () call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() called with valid parameters returned an incorrect code.
Status: Test 19.1 Failed."

Test for Assertion 19.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetChallenge().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1902.

3. (Pre) There exists a target container on the connected card with AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 19.2".

5. Make a challenge1900 == gscBsiGetChallenge() call to the SPS, using
   - hCard /= hCard1902
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.19.2.1 using
   - hCard == hCard1902.

   Print
   "gscBsiGetChallenge() called with a bad handle has been verified.
   Status: Test 19.2 Passed."

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.

Status: Test 19.2 Not Supported."

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() called with a bad handle returned an incorrect code.
Status: Test 19.2 Failed."

Test for Assertion 19.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetChallenge().
2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1903.
3. (Pre) There exists a target container on the connected card with AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).
4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).
5. Make a challenge1900 == gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1903
   - AID == _badGSCAID (GSC) or _badCACAID (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.19.3.1 using
   - hCard == hCard1903.

   Print
   "gscBsiGetChallenge() called with a bad AID value has been verified.
   Status: Test 19.3 Passed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.3 Not Supported."

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() called with a bad AID value returned an incorrect code.
Status: Test 19.3 Failed."

Test for Assertion 19.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 19.5

The method is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetChallenge().
2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1905.
3. (Pre) There exists a target container on the connected card with AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).
4. (Pre) Remove the connected card from the reader.
5. (Pre) Print "Testing of Assertion 19.5".
6. Make a challenge1900 == gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1905
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGetChallenge() called with the connected card removed has been verified.
   Status: Test 19.5 Passed."

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.
Status: Test 19.5 Not Supported."

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() called with the connected card removed returned an incorrect code.
Status: Test 19.5 Failed."

20. gscBsiSkiInternalAuthenticate()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator2000 with one element, the BSIAuthenticator object BSIAuthenticator2000. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. cryptogram2000 is an array of bytes.

Test for Assertion 20.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2001.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Print "Testing of Assertion 20.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2001
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator2000.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.1 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.1.

6. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2001
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID1

- challenge == _goodChallenge.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then cryptogram2000 == an array of bytes containing the symmetric key cryptogram returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_OK, then manually inspect the array cryptogram2000.
   Case 1.1: If cryptogram2000 contains a valid symmetric key cryptogram, then print
   "gscBsiSkiInternalAuthenticate() called with valid parameters has been verified by inspection.
   Status: Test 20.1 Passed."

   Case 1.2: If cryptogram2000 does not contain a valid symmetric key cryptogram, then print
   "gscBsiSkiInternalAuthenticate() called with valid parameters has not been verified by inspection.
   Status: Test 20.1 Failed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.1 Not Supported."

   Case 3: If the gscBsiSkiInternalAuthenticate () call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() called with valid parameters returned an incorrect code.
   Status: Test 20.1 Failed."

Test for Assertion 20.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2002.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Print "Testing of Assertion 20.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2002
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator2000.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.2 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.2.

6. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard /= hCard2002
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID1
   - challenge == _goodChallenge.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.20.2.1 using
   - hCard == hCard2002.

   Print
   "gscBsiSkiInternalAuthenticate() called with a bad handle has been verified.
   Status: Test 20.2 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.2 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() called with a bad handle returned an incorrect code.
Status: Test 20.2 Failed."

Test for Assertion 20.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2003.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) There does not exist a container on the connected card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 20.3".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2003
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator2000.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.3 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.3.

7. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2003
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - algoID == _goodAlgoID1

- challenge == _goodChallenge.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.20.3.1 using
   - hCard == hCard2003.

   Print
   "gscBsiSkiInternalAuthenticate() called with a bad AID value has been verified.
   Status: Test 20.3 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.3 Not Supported."

   Case 2: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() called with a bad AID value returned an incorrect code.
   Status: Test 20.3 Failed."

Test for Assertion 20.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 20.5

The method is tested using a bad parameter.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2005.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Print "Testing of Assertion 20.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2005
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator2000.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.5 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.5.

6. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2005
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID1
   - challenge == _badChallenge.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_PARAM) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.20.5.1 using
   - hCard == hCard2005.

   Print
   "gscBsiSkiInternalAuthenticate() with a bad challenge parameter has been verified.
   Status: Test 20.5 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.5 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() with a bad parameter returned an incorrect code.
Status: Test 20.5 Failed."

Test for Assertion 20.6

The method is tested using a bad cryptographic algorithm identifier.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2006.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Print "Testing of Assertion 20.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2006
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - strctAuthenticator == strctAuthenticator2000.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.6 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.6.

6. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2006
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID2
   - challenge == _goodChallenge.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_ALGO_ID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_ALGO_ID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_ALGO_ID, then:

   Perform Test for Assertion 9.20.6.1 using
   - hCard == hCard2006.

   Print
   "gscBsiSkiInternalAuthenticate() with a bad cryptographic algorithm identifier has been verified.
   Status: Test 20.6 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.6 Not Supported."

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_ALGO_ID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() with a bad cryptographic algorithm identifier returned an incorrect code.
   Status: Test 20.6 Failed."

Test for Assertion 20.7

The method is tested using a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2007.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Print "Testing of Assertion 20.7".

5. (Pre) Establish an authenticated session with the target container on the card:

Make a gscBsiUtilAcquireContext() call to the SPS, using
- hCard == hCard2007
- AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
- strctAuthenticator == strctAuthenticator2000.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 20.7 of gscBsiSkiInternalAuthenticate() cannot be tested".
End Test for Assertion 20.7.

6. Remove the connected card from the reader.

7. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2007
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID1
   - challenge == _goodChallenge.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGetChallenge() called with the connected card removed has been verified.
   Status: Test 20.7 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.7 Not Supported."

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() called with the connected card removed returned an incorrect code.
   Status: Test 20.7 Failed."

Test for Assertion 20.8

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2008.

3. (Pre) There exists a target SKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

4. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2008
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC).

5. (Pre) Print "Testing of Assertion 20.8".

6. Make a cryptogram2000 == gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2008
   - AID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - algoID == _goodAlgoID1
   - challenge == _goodChallenge.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.20.8.1 using
   - hCard == hCard2008.

   Print
   "gscBsiSkiInternalAuthenticate() called without fulfilling the applicable ACR has been verified.
   Status: Test 20.8 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.8 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print

"gscBsiSkiInternalAuthenticate() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 20.8 Failed."

21. gscBsiPkiCompute()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator2100 with one element, the BSIAuthenticator object BSIAuthenticator2100. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. result2100 is an array of bytes.

Test for Assertion 21.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2101.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2101
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.1 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.1.

6. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2101
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID2

- message == _goodMessage.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then result2100 == an array of bytes containing the signature returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_OK, then manually inspect the array result2100.
   Case 1.1: If result2100 contains a valid signature, then print
   "gscBsiPkiCompute() called with valid parameters has been verified by inspection.
   Status: Test 21.1 Passed."

   Case 1.2: If result2100 does not contain a valid signature, then print
   "gscBsiPkiCompute() called with valid parameters has not been verified by inspection.
   Status: Test 21.1 Failed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.1 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() called with valid parameters returned an incorrect code.
   Status: Test 21.1 Failed."

Test for Assertion 21.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2102.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:

- the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2102
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.2 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.2.

6. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard /= hCard2102
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID2
   - message == _goodMessage.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.21.2.1 using
   - hCard == hCard2102.

   Print
   "gscBsiPkiCompute() called with a bad handle has been verified.
   Status: Test 21.2 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.2 Not Supported."

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() called with a bad handle returned an incorrect code.
Status: Test 21.2 Failed."

Test for Assertion 21.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2103.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) There does not exist a container on the connected card with the AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 21.3".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2103
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.3 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.3.

7. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2103
   - AID == _badGSCAID (GSC) or _badCACAID (CAC)
   - algoID == _goodAlgoID2
   - message == _goodMessage.

Verification Goal:
To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.21.3.1 using
   - hCard == hCard2103.

   Print
   "gscBsiPkiCompute() called with a bad AID value has been verified.
   Status: Test 21.3 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.3 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() called with a bad AID value returned an incorrect code.
   Status: Test 21.3 Failed."

Test for Assertion 21.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 21.5

The method is tested using a bad parameter.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2105.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN

- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2105
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.5 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.5.

6. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2105
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID1
   - message == _badMessage.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_PARAM) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.21.5.1 using
   - hCard == hCard2105.

   Print
   "gscBsiPkiCompute() called with a bad parameter has been verified.
   Status: Test 21.5 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.5 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print

"gscBsiPkiCompute() called with a bad parameter returned an incorrect code.
Status: Test 21.5 Failed."

Test for Assertion 21.6

The method is tested using a bad cryptographic algorithm identifier.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2106.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2106
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.6 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.6.

6. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2106
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID1
   - message == _goodMessage.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_ALGO_ID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_ALGO_ID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_ALGO_ID, then:

   Perform Test for Assertion 9.21.6.1 using
   - hCard == hCard2106.

   Print
   "gscBsiPkiCompute() called with a bad cryptographic algorithm identifier has been verified.
   Status: Test 21.6 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.6 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_ALGO_ID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() called with a bad cryptographic algorithm identifier returned an incorrect code.
   Status: Test 21.6 Failed."

Test for Assertion 21.7

The method is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2107.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2107
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2100.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
"A session cannot be established. Assertion 21.7 of gscBsiPkiCompute() cannot be tested".
End Test for Assertion 21.7.

6. (Pre) Remove the connected card from the reader.

7. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2107
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID2
   - message == _goodMessage.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiPkiCompute() called with the connected card removed has been verified.
   Status: Test 21.7 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.7 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() called with the connected card removed returned an incorrect code.
   Status: Test 21.7 Failed."

Test for Assertion 21.8

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2108.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:

- the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

7. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2108
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 21.8".

5. Make a result2100 == gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2108
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - algoID == _goodAlgoID2
   - message == _goodMessage.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.21.8.1 using
   - hCard == hCard2108.

   Print
   "gscBsiPkiCompute() without fulfilling the applicable ACR has been verified.
   Status: Test 21.8 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.8 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 21.8 Failed."

22. gscBsiPkiGetCertificate()

Starting State for Each Test:

1. There exists a Vector strctAuthenticator2200 with one element, the BSIAuthenticator object BSIAuthenticator2200. This object has fields
   - accessMethodType == BSI_AM_PIN
   - keyIDOrReference == _keyIDOrReference1
   - authValue == _goodAuthValue1.

2. certificate2200 is an array of bytes.

Test for Assertion 22.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard2201.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiGetCertificate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 22.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2201
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2200.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.1 of gscBsiPkiGetCertificate() cannot be tested".
   End Test for Assertion 22.1.

6. Make a certificate2200 == gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2201
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then certificate2200 == an array of bytes containing the certificate returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_OK, then manually inspect the array certificate2200.

   Case 1.1: If certificate2200 contains a valid certificate, then print
   "gscBsiPkiGetCertificate() called with valid parameters has been verified by inspection.
   Status: Test 22.1 Passed."

   Case 1.2: If certificate2200 does not contain a valid certificate, then print
   "gscBsiPkiGetCertificate() called with valid parameters has not been verified by inspection.
   Status: Test 22.1 Failed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.1 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate () call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() called with valid parameters returned an incorrect code.
   Status: Test 22.1 Failed."

Test for Assertion 22.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard2202.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiGetCertificate() service has the value BSI_ACR_PIN

- the value of the PIN is _PIN
- the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 22.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2202
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2200.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.2 of gscBsiPkiGetCertificate() cannot be tested".
   End Test for Assertion 22.2.

6. Make a certificate2200 == gscBsiPkiGetCertificate() call to the SPS, using
   - hCard /= hCard2202
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.22.2.1 using
   - hCard == hCard2202.

   Print
   "gscBsiPkiGetCertificate() called with a bad handle has been verified.
   Status: Test 22.2 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.2 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print

"gscBsiPkiGetCertificate() called with a bad handle returned an incorrect code.
Status: Test 22.2 Failed."

Test for Assertion 22.3

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 22.4

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard2204.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiGetCertificate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) There does not exist a container on the connected card with the AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 22.4".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2204
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2200.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.4 of gscBsiPkiGetCertificate() cannot be tested".
   End Test for Assertion 22.4.

7. Make a certificate2200 == gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2204
   - AID == _badGSCAID (GSC) or _badCACAID (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.22.4.1 using
   - hCard == hCard2204.

   Print
   "gscBsiPkiGetCertificate() called with a bad AID value has been verified.
   Status: Test 22.4 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.4 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() called with a bad AID value returned an incorrect code.
   Status: Test 22.4 Failed."

Test for Assertion 22.5

The method is tested with a removed card.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard2205.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiGetCertificate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 22.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2205
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator2200.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If gscBsiUtilAcquireContext() does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.5 of gscBsiPkiGetCertificate() cannot be tested".
   End Test for Assertion 22.5.

6. (Pre) Remove the connected card from the reader.

7. Make a certificate2200 == gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2205
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiPkiGetCertificate() called with the connected card removed has been verified.
   Status: Test 22.5 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.5 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() called with the connected card removed returned an incorrect code.
   Status: Test 22.5 Failed."

Test for Assertion 22.6

The method is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS service is in a reader, connected with handle hCard2206.

3. (Pre) There exists a target PKI provider container on the connected card with the following properties:
   - the ACR for the gscBsiPkiGetCertificate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2206
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

5. (Pre) Print "Testing of Assertion 22.6".

6. Make a certificate2200 == gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2206
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_ACCESS_DENIED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.22.6.1 using
   - hCard == hCard2206.

   Print
   "gscBsiPkiGetCertificate() without fulfilling the applicable ACR has been verified.
   Status: Test 22.6 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.6 Not Supported."

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print

"gscBsiPkiGetCertificate() without fulfilling the applicable ACR returned an incorrect code.
Status: Test 22.6 Failed."

23. gscBsiGetCryptoProperties()

Starting state for each Test:

1. There exists a CryptoProperties object cryptoProps2300 with fields
   - CRYPTOacr strctCRYPTOacr2300, with fields
     - BSIAcr getChallengeACR2300, with fields
       - int getChallengeACRType2300
       - int[] getChallengeKeyIDOrReference2300
       - int getChallengeAuthNb2300
       - int getChallengeACRID2300
     - BSIAcr internalAuthenticateACR2300, with fields
       - int internalAuthenticateACRType2300
       - int[] internalAuthenticateKeyIDOrReference2300
       - int internalAuthenticateAuthNb2300
       - int internalAuthenticateACRID2300
     - BSIAcr pkiComputeACR2300, with fields
       - int pkiComputeACRType2300
       - int[] pkiComputeKeyIDOrReference2300
       - int pkiComputeAuthNb2300
       - int pkiComputeACRID2300
     - BSIAcr createACR2300, with fields
       - int createACRType2300
       - int[] createKeyIDOrReference2300
       - int createAuthNb2300
       - int createACRID2300
     - BSIAcr deleteACR2300, with fields
       - int deleteACRType2300
       - int[] deleteKeyIDOrReference2300
       - int deleteAuthNb2300
       - int deleteACRID2300
     - BSIAcr readTagListACR2300, with fields
       - int readTagListACRType2300
       - int[] readTagListKeyIDOrReference2300
       - int readTagListAuthNb2300
       - int readTagListACRID2300
     - BSIAcr readValueACR2300, with fields
       - int readValueACRType2300
       - int[] readValueKeyIDOrReference2300
       - int readValueAuthNb2300
       - int readValueACRID2300
     - BSIAcr updateValueACR2300, with fields
       - int updateValueACRType2300
       - int[] updateValueKeyIDOrReference2300
       - int updateValueAuthNb2300
       - int updateValueACRID2300
   - int keyLen2300.

Test for Assertion 23.1

The method is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2301.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 23.1".

5. Make a cryptoProps2300 == gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2301
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK (no BSIException is thrown) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

2. If the return code is BSI_OK, then the variables of strctCRYPTOacr2300 are correctly set to indicate access control conditions for all operations.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_OK, then:
   Case 1.1: If
   - each of getChallengeACRType2300, internalAuthenticateACRType2300, pkiComputeACRType2300, createACRType2300, deleteACRType2300, readTagListACRType2300, readValueACRType2300, and updateValueACRType2300 == BSI_ACR_PIN

   and
   - the content of each of getChallengeKeyIDOrReference2300, internalAuthenticateKeyIDOrReference2300, pkiComputeKeyIDOrReference2300,

createKeyIDOrReference2300,
deleteKeyIDOrReference2300,
readTagListKeyIDOrReference2300,
readValueKeyIDOrReference2300, and
updateKeyIDOrReference2300 == _keyIDOrReference1

and

- each of getChallengeAuthNB2300,
internalAuthenticateAuthNB2300, pkiComputeAuthNB2300,
createAuthNB2300, deleteAuthNB2300,
readTagListAuthNB2300, readValueAuthNB2300, and
updateValueAuthNB2300 == 1

and

- each of getChallengeACRID2300,
internalAuthenticateACRID2300, pkiComputeACRID2300,
createACRID2300, deleteACRID2300,
readTagListACRID2300, readValueACRID2300, and
updateValueACRID2300 == 0

then print
"gscBsiGetCryptoProperties() called with valid parameters
has been verified.
Status: Test 23.1 Passed."

Case 1.2: If

- any of getChallengeACRType2300,
internalAuthenticateACRType2300, createACRType2300,
pkiComputeACRType2300, deleteACRType2300,
readTagListACRType2300, readValueACRType2300, and
updateValueACRType2300 /= BSI_ACR_PIN

or

- the content of any of
getChallengeKeyIDOrReference2300,
internalAuthenticateKeyIDOrReference2300,
pkiComputeKeyIDOrReference2300,
createKeyIDOrReference2300,
deleteKeyIDOrReference2300,
readTagListKeyIDOrReference2300,
readValueKeyIDOrReference2300, and
updateKeyIDOrReference2300 /= _keyIDOrReference1

or

- any of getChallengeAuthNB2300,
internalAuthenticateAuthNB2300, pkiComputeAuthNB2300,
createAuthNB2300, deleteAuthNB2300,
readTagListAuthNB2300, readValueAuthNB2300, and
updateValueAuthNB2300 /= 1

or

- any of getChallengeACRID2300,
internalAuthenticateACRID2300, createACRID2300,
pkiComputeACRID2300, deleteACRID2300,
readTagListACRID2300, readValueACRID2300, and
updateValueACRID2300 /= 0

then print
"gscBsiGetCryptoProperties() called with valid parameters
has not been verified.
Status: Test 23.1 Failed."

Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetCryptoProperties() is not supported.
Status: Test 23.1 Not Supported."

Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetCryptoProperties() called with valid parameters returned an incorrect code.
Status: Test 23.1 Failed."

Test for Assertion 23.2

The method is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2302.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Print "Testing of Assertion 23.2".

5. Make a containerProps2300 == gscBsiGetCryptoProperties() call to the SPS, using
   - hCard /= hCard2302
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.23.2.1 using
   - hCard == hCard2302.

   Print
   "gscBsiGetCryptoProperties() called with a bad handle has been verified.
   Status: Test 23.2 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.2 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with a bad handle returned an incorrect code.
   Status: Test 23.2 Failed."

Test for Assertion 23.3

The method is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2303.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) There does not exist a container on the connected card with the AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 23.3".

6. Make a containerProps2300 == gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2303
   - AID == _badGSCAID (GSC) or _badCACAID (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_AID) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.23.3.1 using
   - hCard == hCard2302.

   Print
   "gscBsiGetCryptoProperties() called with a bad AID value has been verified.
   Status: Test 23.3 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.3 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with a bad AID value returned an incorrect code.
   Status: Test 23.3 Failed."

Test for Assertion 23.4

The method is tested with another application having established a transaction lock.

Note: Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 23.5

The method is tested with a removed card.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance with the GSC-IS is in a reader, connected with handle hCard2305.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - the content of the keyID or reference array == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == 0
   - the container is represented by the AID value == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

4. (Pre) Remove the connected card from the reader.

5. (Pre) Print "Testing of Assertion 23.5".

6. Make a containerProps2300 == gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2305
   - AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED (a BSIException is thrown, with BSIException.getErrorCode returning BSI_CARD_REMOVED) or the return code BSI_NO_CARDSERVICE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_NO_CARDSERVICE).

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGetCryptoProperties() called with the connected card removed has been verified.
   Status: Test 23.5 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.5 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with the connected card removed returned an incorrect code.
   Status: Test 23.5 Failed."

Appendix A

List of Symbolic Constants

_badAuthValue == {'B','A','D',' ','A','U','T','H'}

_badAuthValueLen == 8

_badCACAID == "A0000000793333"

_badCACAIDLen == 14

_badCardCommand == {1}

_badChallenge == null

_badChallengeLen == 0

_badGSCAID == "A0000001163333"

_badGSCAIDLen == 14

_badMessage == null

_badMessageLen == 0

_badReaderName == "Bad Reader Name"

_badReaderNameLen == 15

_existingDvalueFull == $\{F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16}\}$

_existingDvalueFullLen == 10

_existingDvalueLen == 4

_existingDvalue1309 == {0,0,0,1}

_existingDvalue1310 == {0,0,0,2}

_existingDvalue1401 == {0,0,0,3}

_existingDvalue1402 == {0,0,0,4}

_existingDvalue1403 == {0,0,0,5}

_existingDvalue1404 == {0,0,0,6}

_existingDvalue1405 == {0,0,0,7}

_existingDvalue1406 == {0,0,0,8}

_existingDvalue1407 == {0,0,0,9}

_existingDvalue1408 == $\{0,0,0,A_{16}\}$

_existingDvalue1701 == {0,0,0,B_{16}}

_existingDvalue1702 == {0,0,0,C_{16}}

_existingDvalue1703 == {0,0,0,D_{16}}

_existingDvalue1704 == {0,0,0,E_{16}}

_existingDvalue1706 == {0,0,0,F_{16}}

_existingDvalue1707 == {0,0,0,G_{16}}

_existingDvalue1709 == {0,0,0,H_{16}}

_existingDvalue1710 == {0,0,1,0}

_existingDvalue1801 == {0,0,1,1}

_existingDvalue1802 == {0,0,1,2}

_existingDvalue1803 == {0,0,1,3}

_existingDvalue1804 == {0,0,1,4}

_existingDvalue1805 == {0,0,1,5}

_existingDvalue1806 == {0,0,1,6}

_existingDvalue1807 == {0,0,1,7}

_existingDvalue1808 == {0,0,1,8}

_existingDvalue1809 == {0,0,1,9}

_existingTagFull == 1

_existingTag1309 == 2

_existingTag1310 == 3

_existingTag1401 == 4

_existingTag1402 == 5

_existingTag1403 == 6

_existingTag1404 == 7

_existingTag1405 == 8

_existingTag1406 == 9

_existingTag1407 == 10

_existingTag1408 == 11

_existingTag1701 == 12

_existingTag1702 == 13

_existingTag1703 == 14

_existingTag1704 == 15

_existingTag1706 == 16

_existingTag1707 == 17

_existingTag1709 == 18

_existingTag1710 == 19

_existingTag1801 == 20

_existingTag1802 == 21

_existingTag1803 == 22

_existingTag1804 == 23

_existingTag1805 == 24

_existingTag1806 == 25

_existingTag1807 == 26

_existingTag1808 == 27

_existingTag1809 == 28

_goodAlgoID1 == DES3-ECB == 81_{16}

_goodAlgoID2 == RSA_NO_PAD == $A3_{16}$

_goodAuthValue1 == {'1','2','3','4','5','6','7','8'}

_goodAuthValue1Len == 8

_goodAuthValue2 == {'8','7','6','5','4','3','2','1'}

_goodAuthValue2Len == 8

_goodCACAID1 == "A0000000792222"

_goodCACAID1Len == 14

_goodCACAID2 == "A0000000791111"

_goodCACAID2Len == 14

_goodCardCommand == {0,0,A_{16},4,0,3,0,0,0,2,3,F_{16},0,0}

_goodCardCommandLen == 14

```
_goodCardResponse == {{6,2,8,3}, {6,2,8,4}, {6,A16,8,1}, {6,A16,8,2},
      {6,A16,8,6}, {6,A16,8,7}, {9,0,0,0}}

_goodChallenge == {'g','o','o','d',
      ' ','c','h','a','l','l','e','n','g','e'}

_goodGSCAID1 == "A0000001162222"

_goodGSCAID1Len == 14

_goodGSCAID2 == "A0000001161111"

_goodGSCAID2Len == 14

_goodMessage == {'g','o','o','d',' ','m','e','s','s','a','g','e'}

_invalidAlgoID == -1

_invalidTag == 0

_keyIDOrReference1 == 1

_keyIDOrReference2 == 1

_newDvalueLen == 4

_newDvalue1301 == {1,0,0,1}

_newDvalue1302 == {1,0,0,2}

_newDvalue1303 == {1,0,0,3}

_newDvalue1304 == {1,0,0,4}

_newDvalue1305 == {1,0,0,5}

_newDvalue1306 == {1,0,0,6}

_newDvalue1307 == {1,0,0,7}

_newDvalue1308 == {1,0,0,8}

_newDvalue1309 == {1,0,0,9}

_newDvalue1310 == {1,0,0,A16}

_newDvalue1801 == {1,0,0,B16}

_newDvalue1802 == {1,0,0,C16}

_newDvalue1803 == {1,0,0,D16}

_newDvalue1804 == {1,0,0,E16}

_newDvalue1805 == {1,0,0,F16}

_newDvalue1806 == {1,0,1,0}
```

_newDvalue1807 == {1,0,1,1}

_newDvalue1808 == {1,0,1,2}

_newDvalue1809 == {1,0,1,3}

_newTag1301 == 101

_newTag1302 == 102

_newTag1303 == 103

_newTag1305 == 104

_newTag1306 == 105

_newTag1307 == 106

_newTag1308 == 107

_newTag1309 == 108

_newTag1404 == 109

_newTag1406 == 110

_newTag1705 == 111

_newTag1708 == 112

_newTag1806 == 113

_newTag1807 == 114

_PIN == {'1','2','3','4','5','6','7','8'}

_tooBigDvalue == {1,2,3,4,5,6,7,8,9,A_{16},B_{16},C_{16},D_{16},E_{16},F_{16}}

Note: Some of the above symbolic constants are not used in the Java test scenarios. They are used in the C test scenarios, and are included here because some of them are, in turn, needed to completely define the specification, in Appendix B, of cards required for BSI testing.

Appendix B

List of Cards Required for Testing of BSI Implementations

Card 1. Can be used for tests 1-all, 2.1, 2.2, 2.3, 2.4, 2.5(C binding), 3-all, 4-all, 5-all, 7-all, 8-all, 11-all, 12-all, 13-all, 14-all, 15-all, 16-all, 17-all, 18-all, 19-all:

- claims conformance to the GSC-IS
- has a container for which
  - createACR, deleteACR, readTagListACR, readValueACR, and updateValueACR are PIN Protected
  - PIN == _PIN
  - the content of the keyID or reference array == _keyIDOrReference1
  - number of access methods logically combined in the ACR == 1
  - ACRID == 0
  - AID == _goodCACAID1 (CAC) or _goodGSCAID1 (GSC)
  - can accommodate a new data item of length _newDvalueLen
  - there exist the following data items (TLV):
    - {_existingTag1309, 4, _existingDvalue1309}
    - {_existingTag1310, 4, _existingDvalue1310}
    - {_existingTag1401, 4, _existingDvalue1401}
    - {_existingTag1402, 4, _existingDvalue1402}
    - {_existingTag1403, 4, _existingDvalue1403}
    - {_existingTag1404, 4, _existingDvalue1404}
    - {_existingTag1405, 4, _existingDvalue1405}
    - {_existingTag1406, 4, _existingDvalue1406}
    - {_existingTag1407, 4, _existingDvalue1407}
    - {_existingTag1408, 4, _existingDvalue1408}
    - {_existingTag1701, 4, _existingDvalue1701}
    - {_existingTag1702, 4, _existingDvalue1702}
    - {_existingTag1703, 4, _existingDvalue1703}
    - {_existingTag1704, 4, _existingDvalue1704}
    - {_existingTag1706, 4, _existingDvalue1706}
    - {_existingTag1707, 4, _existingDvalue1707}
    - {_existingTag1709, 4, _existingDvalue1709}
    - {_existingTag1710, 4, _existingDvalue1710}
    - {_existingTag1801, 4, _existingDvalue1801}
    - {_existingTag1802, 4, _existingDvalue1802}
    - {_existingTag1803, 4, _existingDvalue1803}
    - {_existingTag1804, 4, _existingDvalue1804}
    - {_existingTag1805, 4, _existingDvalue1805}
    - {_existingTag1806, 4, _existingDvalue1806}
    - {_existingTag1807, 4, _existingDvalue1807}
    - {_existingTag1808, 4, _existingDvalue1808)
    - {_existingTag1809, 4, _existingDvalue1809}
- has a second container for which
  - updateValueACR is PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID2 (CAC) or _goodGSCAID2 (GSC)
  - there exists one data item:

        - {_existingTagFull, 10, _existingDvalueFull}
      which comprises the entire container
- neither container has a data item for which tag ==
  - _newTag1301
  - _newTag1302
  - _newTag1303
  - _newTag1305
  - _newTag1306
  - _newTag1307
  - _newTag1308
  - _newTag1309
  - _newTag1404
  - _newTag1406
  - _newTag1705
  - _newTag1708
  - _newTag1806
  - _newTag1807
- does not have a container for which
  - AID == _badCACAID (CAC) or _badGSCAID (GSC)

Card 2. Can be used for test 2.5 (Java binding), 2.6 (C binding):

- bad card

Card 3. Can be used for tests 20-all, 21-all, 22-all, 23-all:

- claims conformance to the GSC-IS
- has a PKI provider container for which
  - dataCreateACR, dataDeleteACR, readTagListACR, readValueACR, dataUpdateACR, pkiComputeACR, internalAuthenticateACR, and getChallengeACR are PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID1 (CAC) or _goodGSCAID1 (GSC)
- has an SKI provider container for which
  - pkiComputeACR, internalAuthenticateACR, and readValueACR are PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID2 (CAC) or _goodGSCAID2 (GSC)
- does not have a container for which
  - AID == _badCACAID (CAC) or _badGSCAID (GSC)

Substantive changes from the February 5 version of this document